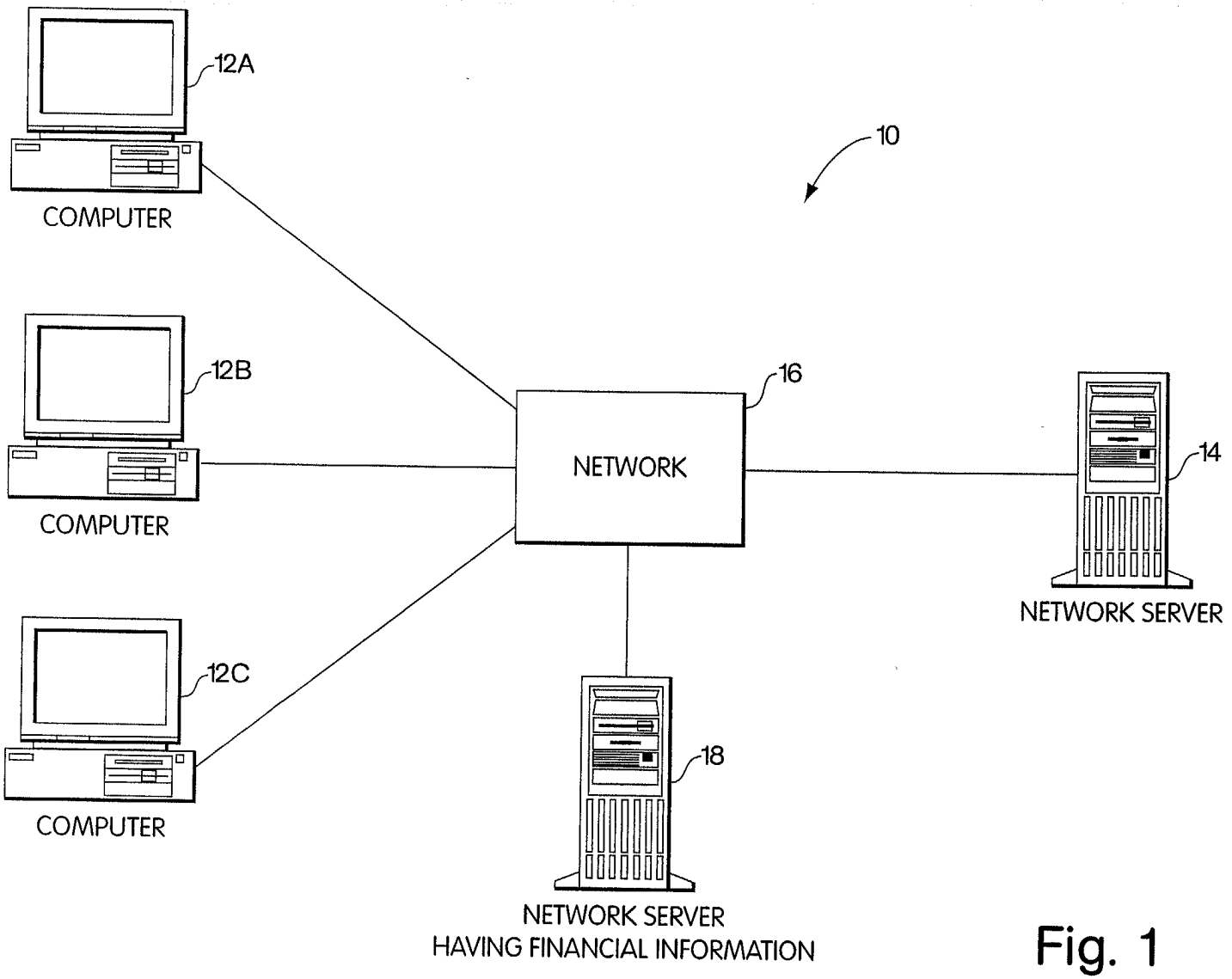

Fig. 1

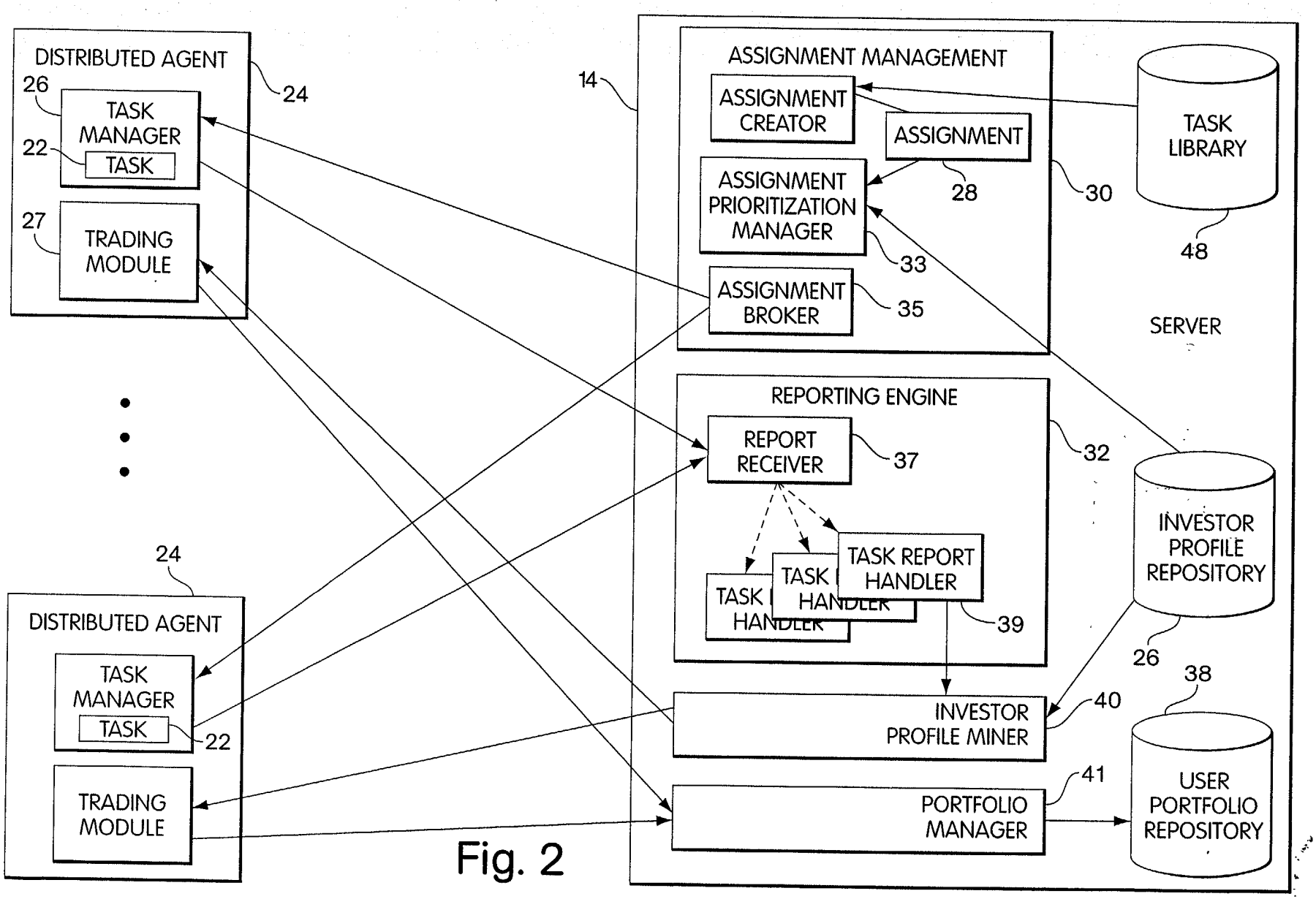

Fig. 2

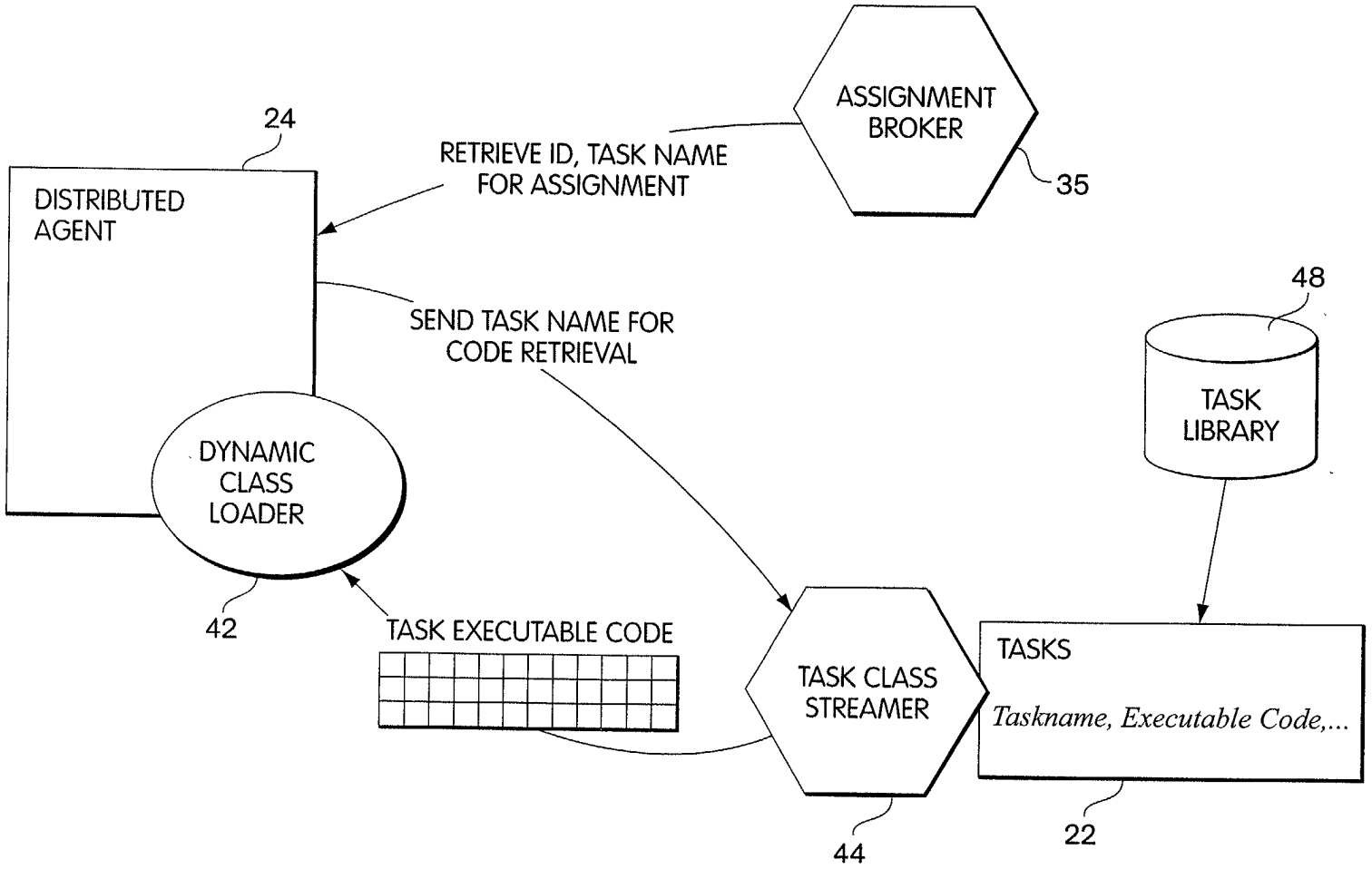
ASSIGNMENT BROKER
35
24
RETRIEVE ID, TASK NAME FOR ASSIGNMENT
DISTRIBUTED AGENT
SEND TASK NAME FOR CODE RETRIEVAL
48
TASK LIBRARY
DYNAMIC CLASS LOADER
42
TASK EXECUTABLE CODE
TASK CLASS STREAMER
TASKS
Taskname, Executable Code,...
44
22

Fig. 3

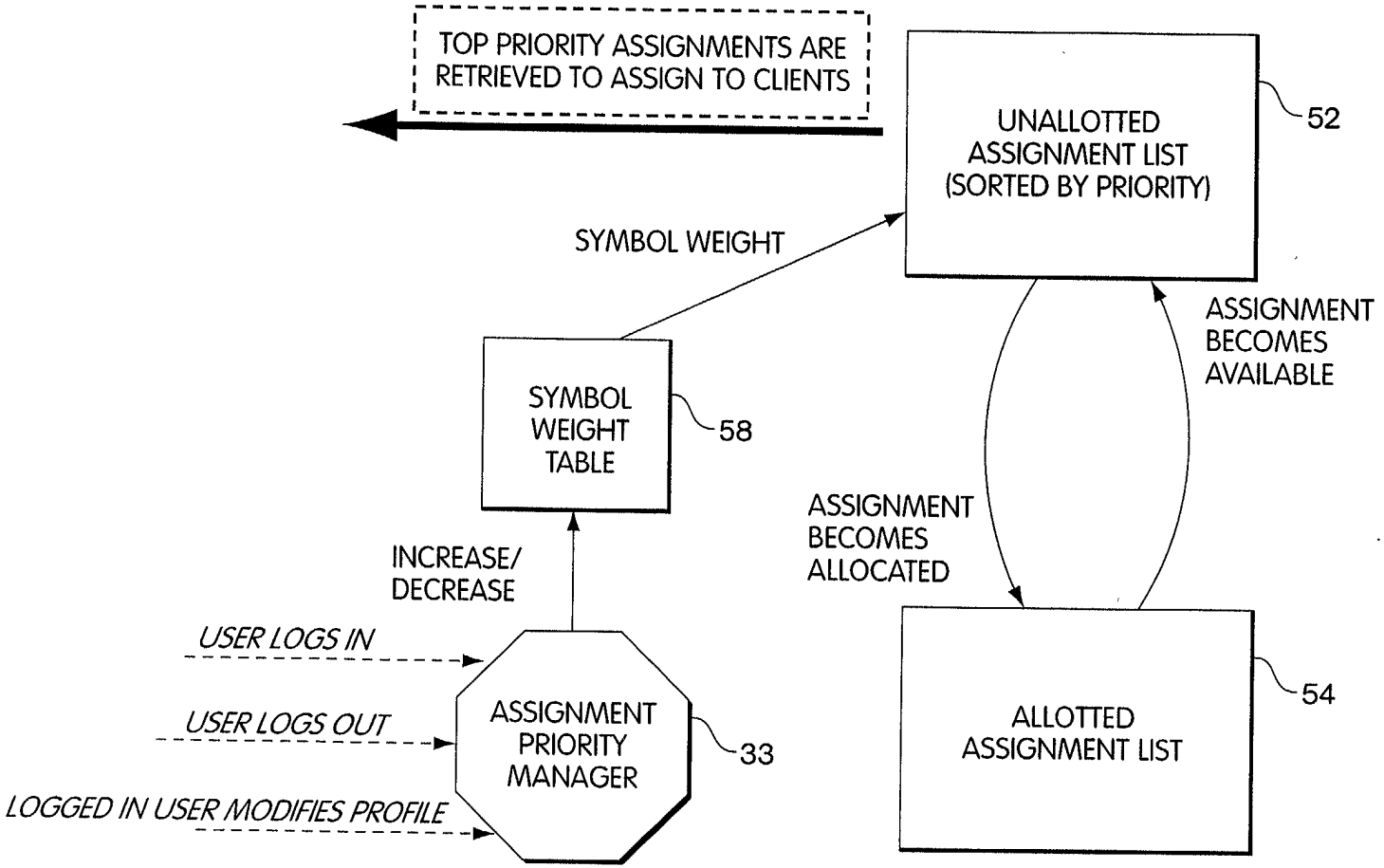
TOP PRIORITY ASSIGNMENTS ARE RETRIEVED TO ASSIGN TO CLIENTS
UNALLOTTED ASSIGNMENT LIST (SORTED BY PRIORITY)
52
SYMBOL WEIGHT
ASSIGNMENT BECOMES AVAILABLE
SYMBOL WEIGHT TABLE
58
ASSIGNMENT BECOMES ALLOCATED
INCREASE/ DECREASE
USER LOGS IN
USER LOGS OUT
LOGGED IN USER MODIFIES PROFILE
ASSIGNMENT PRIORITY MANAGER
33
ALLOTTED ASSIGNMENT LIST
54

Fig. 4

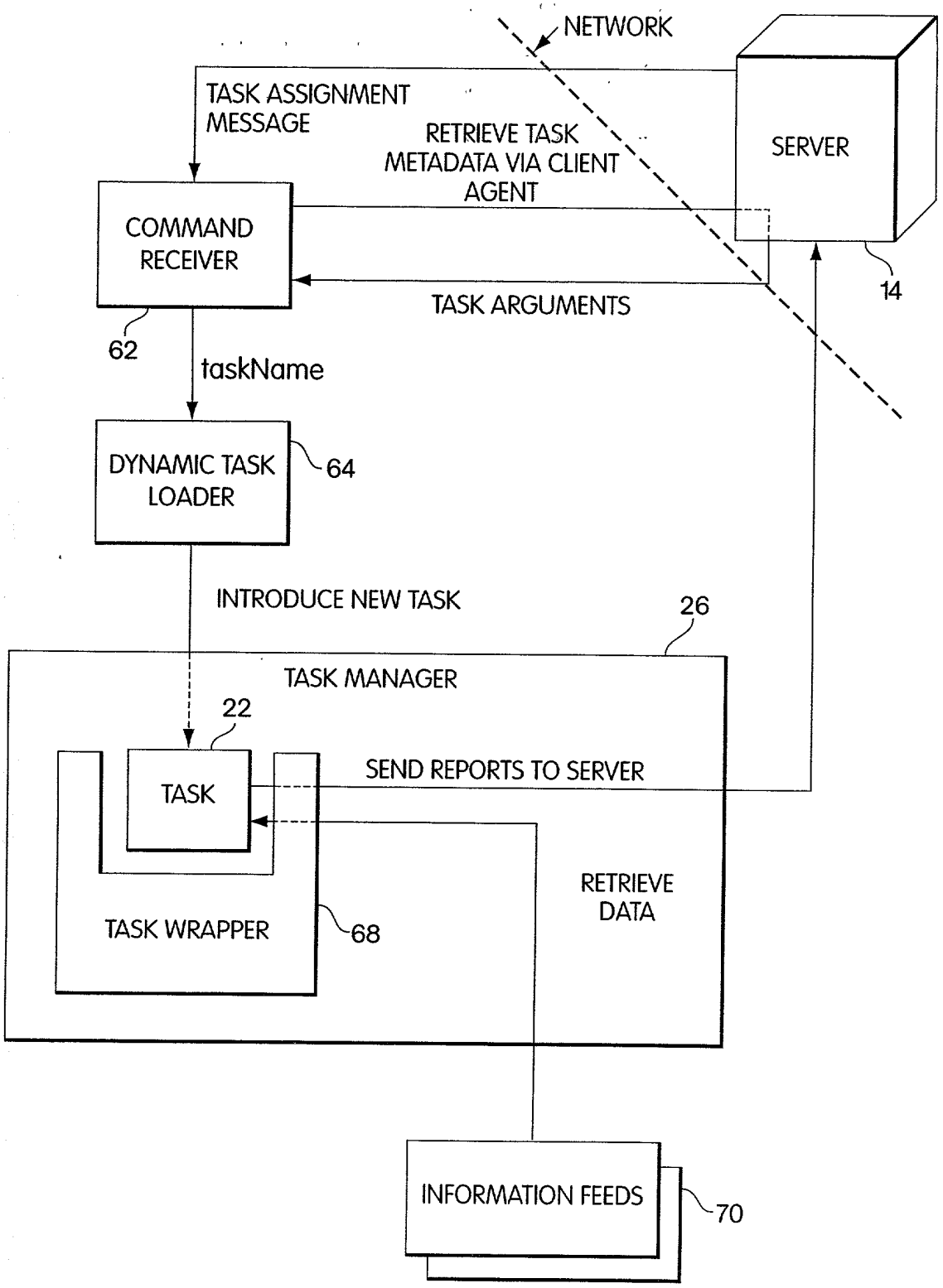
NETWORK
TASK ASSIGNMENT MESSAGE
RETRIEVE TASK METADATA VIA CLIENT AGENT
SERVER
COMMAND RECEIVER
TASK ARGUMENTS
14
62
taskName
DYNAMIC TASK LOADER
64
INTRODUCE NEW TASK
26
TASK MANAGER
22
TASK
SEND REPORTS TO SERVER
RETRIEVE DATA
TASK WRAPPER
68
INFORMATION FEEDS
70

Fig. 5

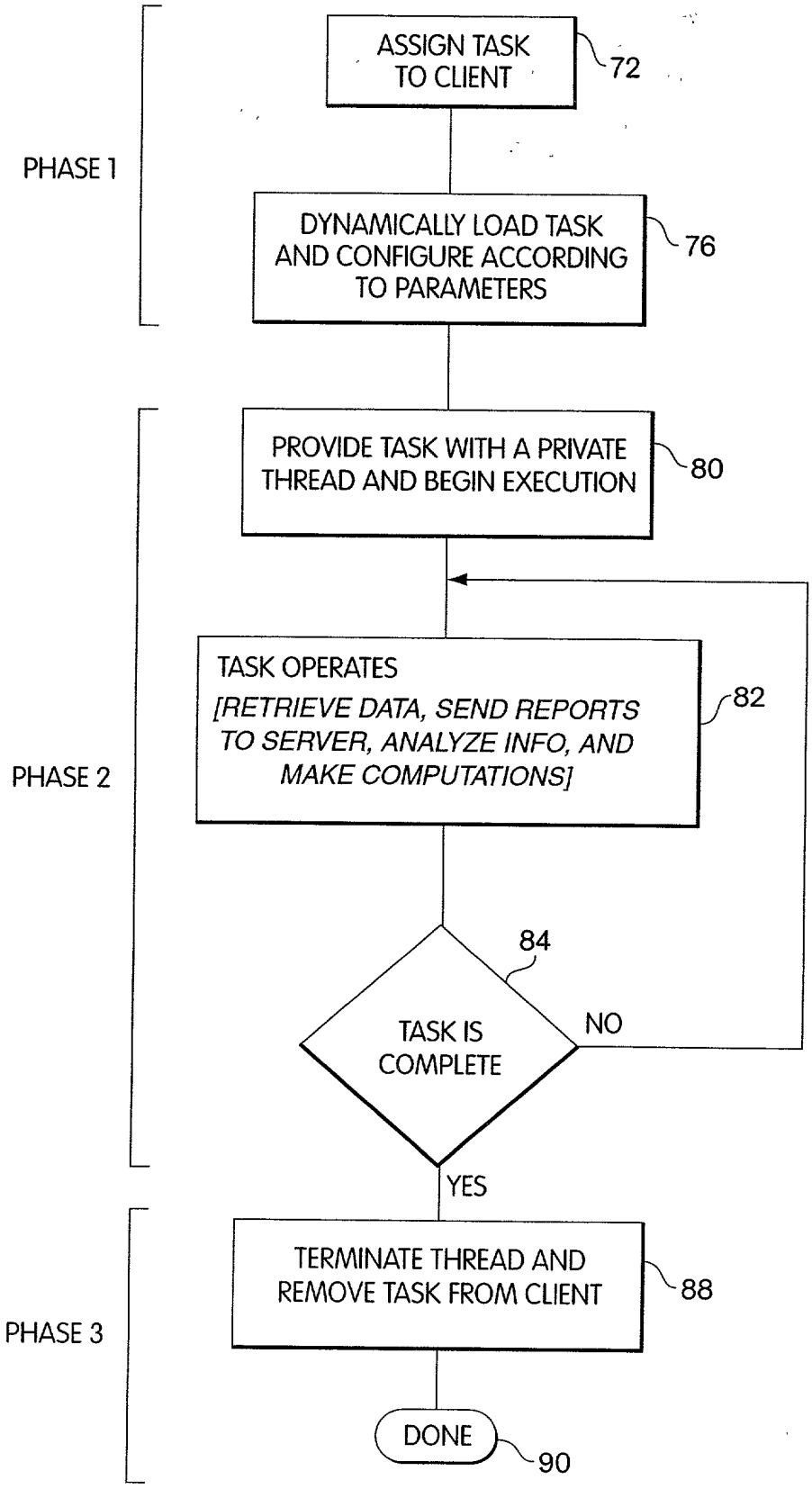
PHASE 1
ASSIGN TASK TO CLIENT
72
DYNAMICALLY LOAD TASK AND CONFIGURE ACCORDING TO PARAMETERS
76
PHASE 2
PROVIDE TASK WITH A PRIVATE THREAD AND BEGIN EXECUTION
80
TASK OPERATES
[RETRIEVE DATA, SEND REPORTS TO SERVER, ANALYZE INFO, AND MAKE COMPUTATIONS]
82
84
TASK IS COMPLETE
NO
YES
PHASE 3
TERMINATE THREAD AND REMOVE TASK FROM CLIENT
88
DONE
90

Fig. 6

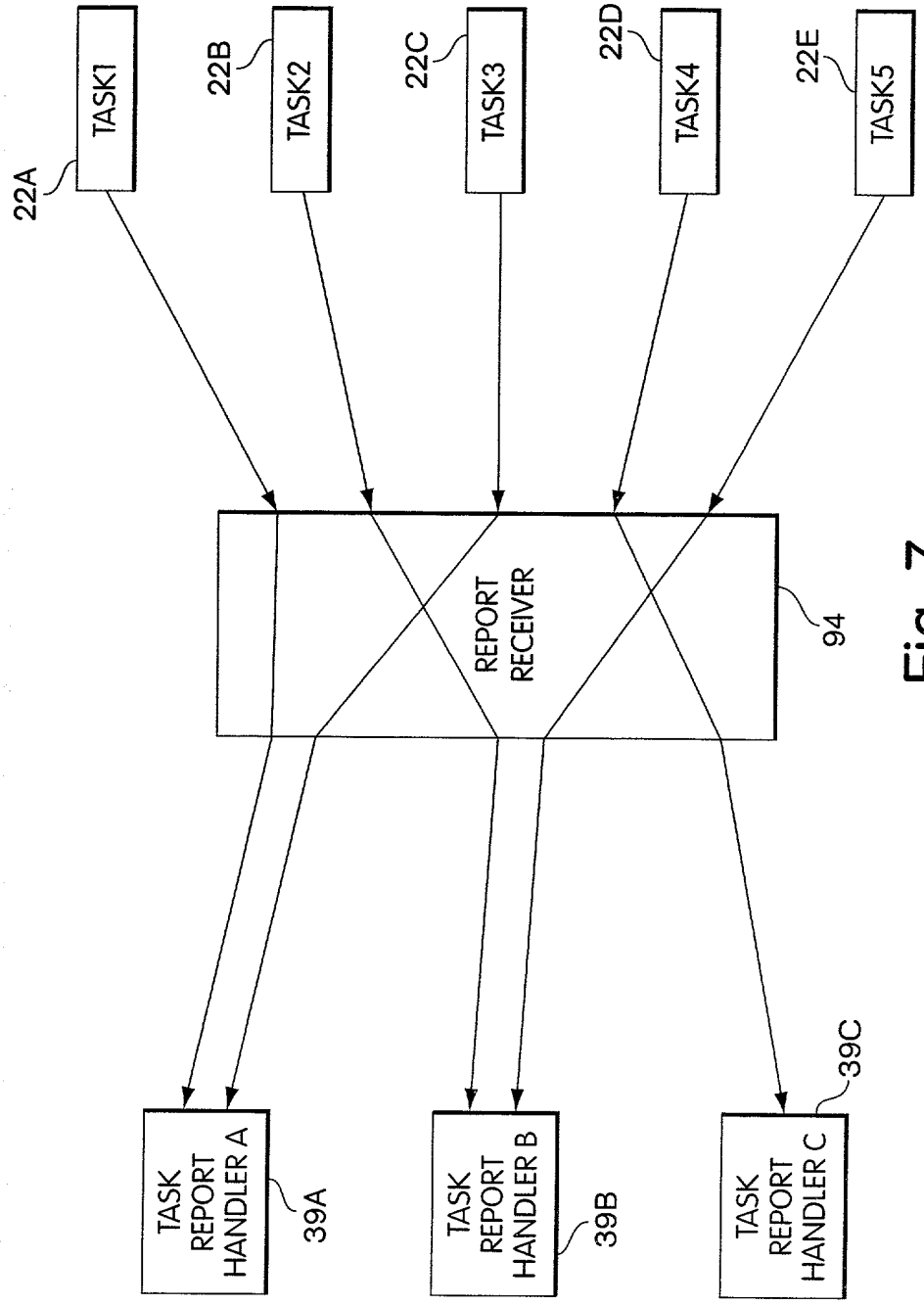
TASK1
TASK2
TASK3
TASK4
TASK5
22A
22B
22C
22D
22E
REPORT RECEIVER
94
TASK REPORT HANDLER A
39A
TASK REPORT HANDLER B
39B
TASK REPORT HANDLER C
39C

Fig. 7

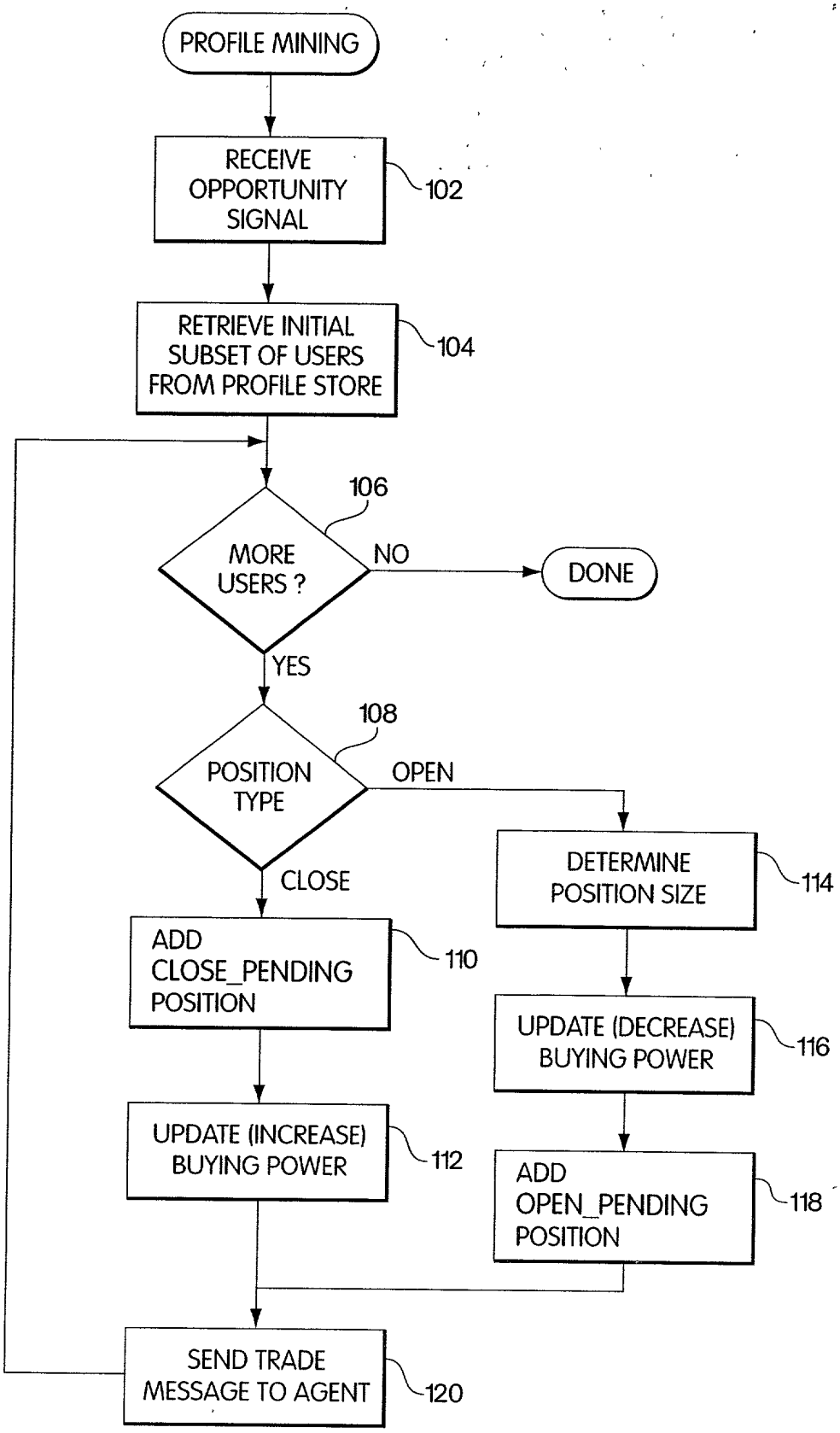
PROFILE MINING
RECEIVE OPPORTUNITY SIGNAL
102
RETRIEVE INITIAL SUBSET OF USERS FROM PROFILE STORE
104
106
MORE USERS ?
NO
DONE
YES
108
POSITION TYPE
OPEN
CLOSE
DETERMINE POSITION SIZE
114
ADD CLOSE_PENDING POSITION
110
UPDATE (DECREASE) BUYING POWER
116
UPDATE (INCREASE) BUYING POWER
112
ADD OPEN_PENDING POSITION
118
SEND TRADE MESSAGE TO AGENT
120

Fig. 8

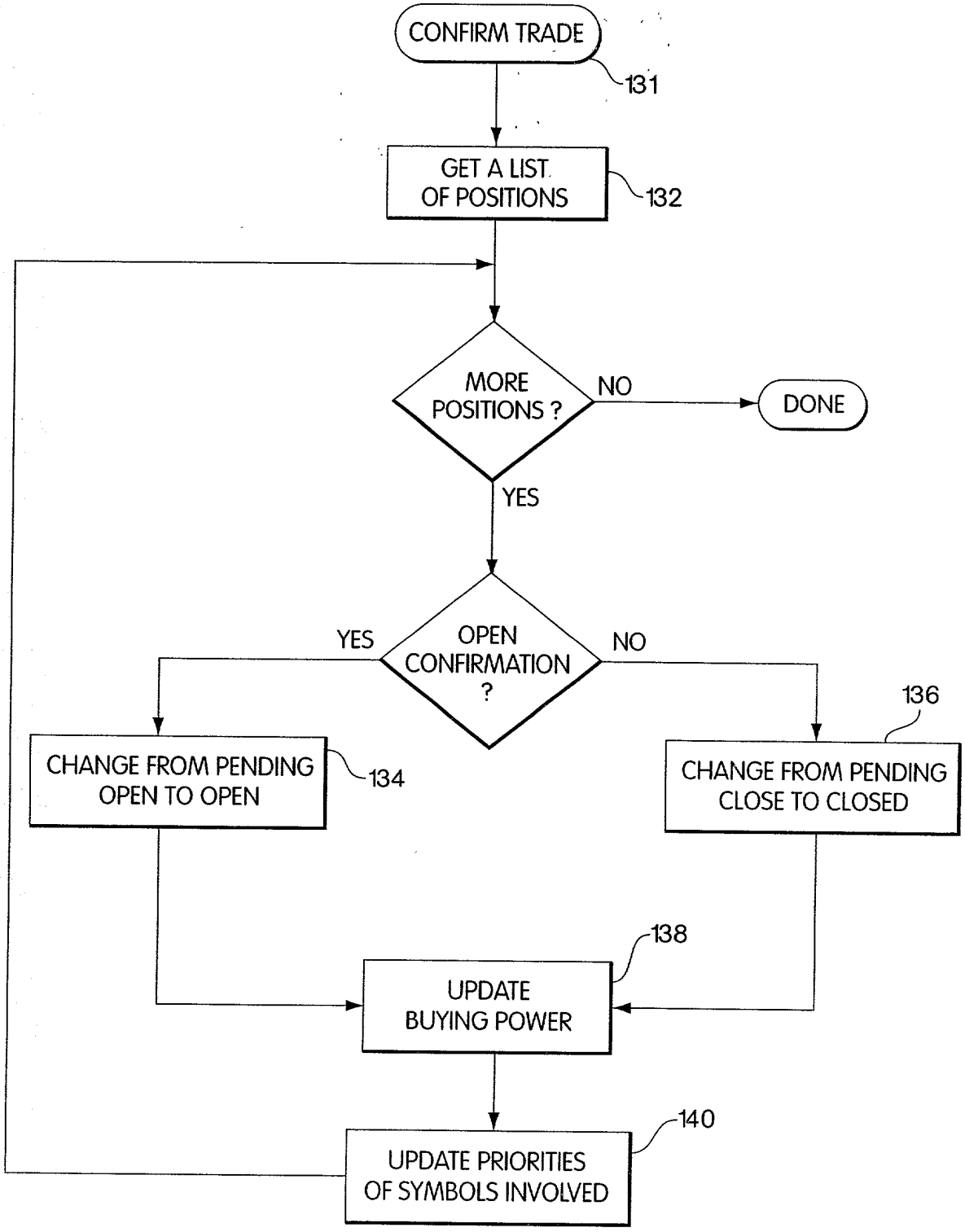
CONFIRM TRADE
131
GET A LIST OF POSITIONS
132
MORE POSITIONS ?
NO
DONE
YES
OPEN CONFIRMATION ?
YES
NO
136
CHANGE FROM PENDING OPEN TO OPEN
134
CHANGE FROM PENDING CLOSE TO CLOSED
138
UPDATE BUYING POWER
140
UPDATE PRIORITIES OF SYMBOLS INVOLVED

Fig. 9

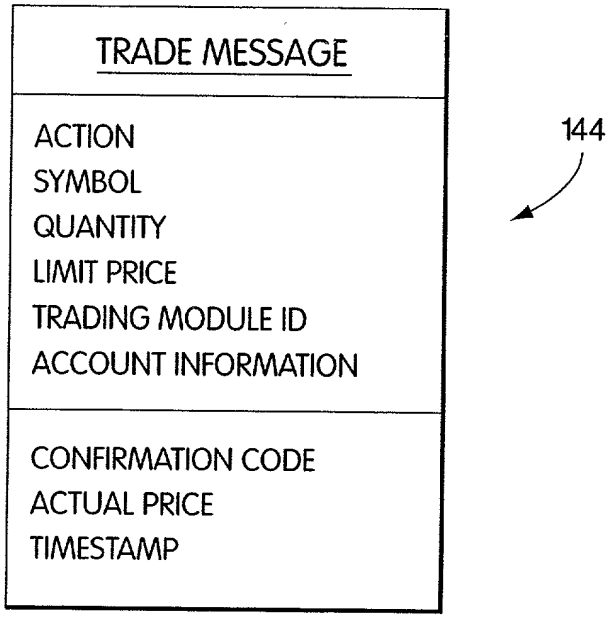

Fig. 10

Investment Profile for <Sample User>

This information is used to customize trading strategies to your needs.

Trade Parameters

Amount Per Trade ($):	Low end of range	High end of range
Shares Per Trade:	Low end of range	High end of range
Price Per Share ($):	Low end of range	High end of range

Equity Parameters

Volatility	Lowest	Low	Medium	High	Highest	All
Company Size	Smallest	Small	Medium	Large	Largest	All
Volume	Lowest	Low	Medium	High	Highest	All
Preferred Indices	DJIA	S&P500	NASDAQ100			

SAVE CANCEL

Fig. 11

```
/**
 * Wrapper class to pass arguments to a task instance on the client.
 */
public class TaskArguments implements java.io.Serializable {

 /**
 * Unique name designating which class this task corresponds to.
 */
public String taskName;

 /**
  * Execution parameters passed to a task when it is instantiated.
  * This usually takes the form of a Hash Table of objects. The structure
  * is flexible to allow different numbers and sizes of
  * parameter to be passed to particular tasks.
  */
 public byte[] argByteArray;
}
```

Fig. 12A

```
/**
 * A collection of services that a task can utilize during its exectution on the
 * client. In order to maintain a high level of modularity, task communication
 * with either the client or server must occur through the methods of this
 * interface.
 */
public interface TaskServiceProvider extends Serializable {

 /**
  * Transmits a report to the server on the wrapped task's request.
  *
  * @param  reportText  the report to be sent
  */
 public void issueReport( String reportText );

 /**
  * Transmits a request for points to the server on the wrapped task's
request.
  */
 public void requestPoints();

 /**
  * Creates and installs a NewsDocReceiver for this task with the specified
feed.
  *
  * @param   task       the concerned task
  * @param   feedKey    describes the feed to use
  */
 public NewsDocReceiver installNewsDocReceiver( String feedKey );

 ////////////////////////////////////////////////////////////////
 //Following are service request that tasks need during execution
 ////////////////////////////////////////////////////////////////

 public Vector getQuotes(Vector symbols) throws SB_Exception;

 public void linkToDataFeed(Observer o, Vector symbols);
 public void unLinkFromDataFeed(Observer o, Vector symbols);

 public Vector getNASDAQTopVolumeLeaders(int num) throws SB_Exception;
 public Vector getNYSETopVolumeLeaders(int num) throws SB_Exception;
 public Vector getAMEXTopVolumeLeaders(int num) throws SB_Exception;

 public Vector getNASDAQTopPercentageLeaders(int num) throws SB_Exception;
 public Vector getNYSETopPercentageLeaders(int num) throws SB_Exception;
 public Vector getAMEXTopPercentageLeaders(int num) throws SB_Exception;

 public Vector getHistoricalData(String symbol, Calendar startDay, Calendar
endDay) throws SB_Exception;
 public boolean checkIfMarketsOpen() throws SB_Exception;
}
```

Fig. 12B

```
/**
 * Provides access to the thread wait and notify methods. This is used when an
 * object that is not the thread owner is running and wants wait/notify control
 * over its thread.
 */
public interface remoteThreadMonitor {

  /**
   * Remote equivalent of Object.wait() .
   */
  public void remoteWait();

  /**
   * Remote equivalent of Object.notifyAll() .
   */
  public void remoteNotifyAll();

}
```

Fig. 12C